Towards a Quality Model for Model Composition Effort

Kleinner Farias[1], Alessandro Garcia[2], Carlos Lucena[2], Luiz Gonzaga Jr[1],
Cristiano André da Costa[1], Rodrigo da Rosa Righi[1], Fábio Basso[3], Toacy Oliveira[3]
[1]PIPCA, University of Vale do Rio dos Sinos, São Leopoldo, Brazil
[2]Informatics Department, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil
[3]COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
kleinnerfarias@unisinos.br, {afgarcia,lucena}@inf.puc-rio.br,
{lgonzaga,cac,rrrighi}@unisinos.br, {fabiopbasso,toacy}@cos.ufrj.br

ABSTRACT
This paper proposes an initial quality model for model composition effort, which serves as a *frame of reference* to developers and researchers to plan and perform qualitative and quantitative investigations, as well as replicate and reproduce empirical studies. A series of empirical studies supports the proposed quality model, including five industrial case studies, two controlled experiments, three quasi-experiments, interviews and seven observational studies. Moreover, these studies have systematically demonstrated the real benefits of using a frame of reference to enable learning about model composition effort from experimentation.

Categories and Subject Descriptors
D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement

General Terms
Measurement, Documentation, Design, Experimentation

Keywords
Model composition, empirical software engineering.

1. INTRODUCTION
Model composition plays a central role in many software engineering activities, e.g. evolving design models to add new features [1] or reconciling multiple models developed in parallel by different software development teams [2]. The composition of design models can be defined as a set of tasks that should be performed over two input models, M_A and M_B, to produce an output intended model, M_{AB}. For this, software developers use the composition techniques to match the model elements in M_A and M_B by automatically "guessing" their semantics and then combine the corresponding elements to create the output intended model, M_{AB}. Nevertheless, the state-of-the-art composition techniques can produce an output composed model, M_{CM}, that does not match with the intended model, M_{AB}, i.e. $M_{CM} \neq M_{AB}$. This is because M_A and M_B often conflict with each other and commonly these conflicts are converted into inconsistencies in M_{CM}. Hence, developers should invest some considerable effort to resolve them, i.e. transforming M_{CM} into M_{AB}. In fact, compose design model is still considered a tedious, error-prone, and time-consuming task [1][2].

However, researchers rarely perform empirical studies concerning model composition or confront the collected results because the current literature fails to provide a "frame of reference" that guides them to produce comparable results. Still, the influential factors of composition effort cannot even be comparable taking into consideration a huge number of confounding variables in real-life contexts. In [3], Runeson and Host emphasize the real need for defining the frame of reference to make the context of empirical studies clear, and helps conducting the research and reviewing the results of them.

Without a frame of reference, it is hard (if not impossible) to: (1) replicate empirical studies as they cannot specify and test the same hypothesis in different analyses or even compare the effects of similar experimental procedures adopted [3]; (2) compare the confidence level for results of an original and replicated study, thereby jeopardizing the improvement of the internal validity and reliability of the conclusions, and hindering the generalization; and (3) generalize results by minimizing the threats to external validity since they are not able to reproduce the design, the planning or even the execution of practical studies.

This paper, therefore, proposes an initial quality model (described in Section 3) for model composition effort, which serves as a *frame of reference* to developers and researchers to plan and perform qualitative and quantitative investigations, as well as replicate and reproduce empirical studies. The proposed quality model is based on an analysis of the current literature [8][10] and stemmed from authors own experiences in conducting qualitative and quantitative research concerning model composition effort, including five industrial case studies [4], two controlled experiments [5][6], three quasi-experiments [7], interviews and seven observational studies [4][5][6].

2. RELATED WORK
Some quality models in the area of modeling have been proposed through the last decades, such as [8][10][11][12][13]. In [10] and [11], the authors present quality models for conceptual modeling. However, they do not convey any concept related to model composition, such as conflicts and inconsistencies. In [8], Lange aims at proposing an extension of [10] and [11] in the context of software modeling; they provide guidelines for selecting metrics and rules to quantify the quality of UML models. The purpose of this quality model is to support a broad quality evaluation of UML models. Although the Lange's quality model has been created based on a literature review and on experiences from industrial

SAC'14, March 24-28, 2014, Gyeongju, Korea.

http://dx.doi.org/10.1145/2554850.2555131

case studies, it is not suitable to evaluate model composition effort due to the reasons described in the previous section.

Thus, this paper overcomes some critical problems so that researchers and developers are able to characterize and evaluate model composition tasks (Section 3.1). The main differences considering the previous studies are (1) an abstract syntax is defined to represent the concepts that are the basis of the quality model, (2) new concepts are included in the model (such as conflict, inconsistency, composition technique, and design characteristic), and (3) four quality notions are added (such as effort, application, detection, and resolution notions).

3. QUALITY MODEL

3.1 Model Composition Effort

Model composition effort refers to the time to produce the output intended model. Figure 1 shows an effort equation that summarizes three complementary facets of composition effort. The equation makes explicit that developers invest some effort to perform three key tasks to produce an intended model, M_{AB}, from two input models M_A, i.e. the base model, and M_B, the model having the changes to be inserted into M_A. It is important to highlight that developers usually need to spend some additional effort to solve inconsistencies in M_{CM} before producing M_{AB}. These three tasks are: (1) $f(M_A,M_B)$, effort to apply composition technique to produce M_{CM} from M_A and M_B; (2) $diff(M_{CM}, M_{AB})$, effort to detect inconsistencies in M_{CM}; and (3) $g(M_{CM})$, the effort to resolve inconsistencies, i.e. the effort to transform M_{CM} into M_{AB}. Note that if M_{CM} is equal to M_{AB}, then $diff(M_{CM},M_{AB}) = 0$ and $g(M_{CM}) = 0$. Otherwise, $diff(M_{CM},M_{AB}) > 0$ and $g(M_{CM}) > 0$. Thus, developers spend effort to accommodate changes from M_B to M_A.

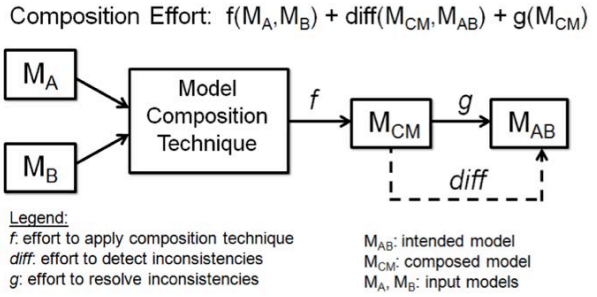

Figure 1. Overview of model composition effort: an equation.

3.2 Abstract Syntax of the Quality Model

Figure 2 shows the abstract syntax of the quality model, which identifies the main concepts and relationship. It follows the UML metamodel specification pattern. The numbers in Figure 2 correspond to the numbers in brackets of the quality notions to be discussed in Section 3.3. Following we described each one of these concepts and relationships.

Domain. This concept represents an area of expertise or application that needs to be examined to solve a problem. The solution of the problem is represented in a design model

Modeling Language. It is the concept that represents the language used to design a software system. Object-oriented modeling languages and aspect-oriented modeling languages are two examples of typical categories of languages used to represent significantly different forms of design decompositions.

Design Model. It refers to the diagram used to represent static and dynamic aspects of a software system. UML class and sequence diagrams are examples of these design models. Developers commonly use these two diagrams, for example, to design structural and dynamic aspects of an application. Moreover, a design model represents the concepts (and their relations) from a domain. This representation helps to describe this domain.

User. It represents a person who interprets design models to get an understanding of the domain [8]. A user can interpret one (or more) design model and compose design models for any particular purpose. Additionally, the user detects and resolves inconsistencies that arise from the compositions. Typical categories of users are software developers and researchers.

Conflict. It is the concept that represents the contradictions between different *Design Models* to be composed. Conflicts arise when the design models have conflicting changes. These contradictions happen when the ordered association *composes: Design Model [2..*]* from *User* to *Design Model* is instantiated. Thus, conflict is a concept derived from the association *composes*. For example, a developer defines that a class is abstract (i.e., *isAbstract* = true) while another developer specifies that the same class is concrete (i.e., *isAbstract* = false). *User* should grasp and deal with these conflicts to produce the intended design model.

Inconsistency. It is the concept that represents the defects found in the output composed model and usually arises because *User* tends to incorrectly resolve a *Conflict*. For example, developers can incorrectly resolve the conflict whether a class should be abstract or not.

Design Characteristic. A design characteristic is the concept that illustrates the strategies used by developers to structure design models, including coupling and cohesion. Design characteristics are used to improve, for example, the capability of design models to be (more straightforwardly) composed. The design characteristics can be also used to indicate error proneness. An example of this design characteristic is the model stability [9][14].

Composition Technique. It is the concept that represents the technique used by developers to compose the design models. Examples of these techniques are Epsilon® and IBM Rational Software Architect®. A model composition technique defines a set of operators that are used to manipulate the input model elements.

3.3 Quality Notions

We propose four quality notions, namely *effort*, *application*, *detection*, and *resolution*, and tailor three other ones from the previous works [8][13], namely *semantic*, *social* and *syntactic*. Using these quality notions researchers can qualitatively evaluate the composition effort in different contexts, as well as compare the results, since they will be based on a common frame of reference that drives the studies. Each quality notion is carefully described as follows.

Syntactic Quality (1). It represents the correctness of design models produced by a design modeling language [8]: if a design modeling language is not properly used, then some syntactic inconsistencies may emerge. This quality notion is relevant to our quality model as syntactic inconsistencies can also arise during model compositions [1]. Developers concern with checking the syntactic consistency of M_{CM}. The degree of correctness should be evaluated in terms of the presence or absence of inconsistencies in M_{CM}. In other words, syntactic quality is computed by measuring the inconsistencies resulting from conflicts between the input

models. This notion helps developers to identify the number of deviations in M_{CM} with respect to the language specification.

Semantic Quality (2). This notion deals with the degree of correspondence between the design model and the problem domain [8]. If the semantics of the model elements are affected, the main purpose of use of the design models, i.e. communication between the team members can be damaged.

Social Quality (3). Design models are essentially used to communicate design decisions between the software development teams. If there is a disagreement between the interpretations of the design models, the communication between the developers is severely impaired. So, researchers should elaborate studies in order to understand the effects of the misinterpretations on the implementation.

Effort Quality (4). It refers to the cost, including time, that developers should invest to produce an output intended model, M_{AB}. It is expected that the practice of applying a composition technique, detecting, and resolving inconsistencies is not an effort-consuming task.

Application Quality (5). It addresses the ease of producing an output composed model by applying a model composition technique. Ideally, developers need to easily compose design models, using composition technique, including heuristic-based or specification-based composition techniques.

Detection Quality (6). When inconsistencies arise, developers should be able to quickly locate them. If the detection of inconsistencies is hard, then the assurance of the correctness of the models may also be hard. Researchers should study the degree of difficulty that developers face to located inconsistency so that consistency in M_{CM} can be assured. The focus of this quality notion is on evaluating the cost to localize inconsistencies in M_{CM}.

Resolution Quality (7). Developers should invest some additional effort trying to find some solution to the inconsistencies located. Otherwise, the practice of composing design model can become prone to inconsistencies or even require more effort than it would be expected. This additional effort can make the practice of assuring the consistency of the composed models difficult and costly. This notion, therefore, addresses the degree of difficulty to resolve inconsistencies.

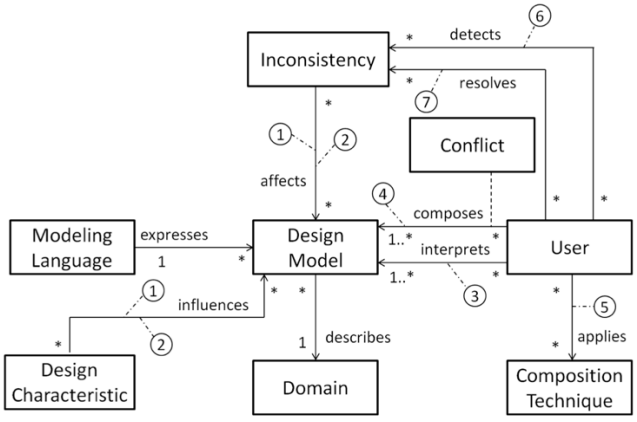

Figure 2. Abstract syntax of the quality model for model composition (based on [8]).

4. CONCLUSIONS AND FUTURE WORK

Researchers and developers recognize the need to evaluate model composition effort. However, the evaluation without any quality model is not trivial, as usually developers have no previous knowledge or experience about empirical evaluations of model composition. This paper addressed an ever-present problem: the difficult of planning and performing qualitative and quantitative investigations, as well as replicating and reproducing empirical studies. Thus, we presented an initial quality model for model composition effort, which serves as a *frame of reference.* To date a systematic description on what factors affect the developers' effort and how they ideally and practically should be evaluated was insufficiently covered in the literature.

Therefore, we can see this work as a first step in a more ambitious agenda to propose a more established and empirically ground frame of reference for evaluation of model composition effort in different real-world contexts. Lastly, we hope that the issues outlined throughout the paper encourage other researchers to perform empirical studies following the proposed quality model and also evaluate it in future under different circumstances.

5. REFERENCES

[1] Mens, T. A State-of-the-Art Survey on Software Merging, IEEE Transac. on Soft. Engineering, 28(5):449-562, 2002.

[2] Rosa, M et. al., Business Process Model Merging: An Approach to Business Process Consolidation, Journal Trans. on Soft. Eng. Method. vol. 22, no. 2, 2013.

[3] Runeson, P., et al., Variation Factors in the Design and Analysis of Replicated Controlled Experiments, Journal Empirical Software Engineering, pages 1-28, 2013.

[4] Farias, K., Garcia, A., Whittle, J., Lucena, C., Analyzing the Effort of Composing Design Models of Large-Scale Software in Industrial Case Studies, In: MODELS'13, 2013.

[5] Farias, K., Garcia, A., Lucena, C., Evaluating the Impact of Aspects on Inconsistency Detection Effort: A Controlled Experiment, In: MODELS'12, pages 219-234, 2012.

[6] Farias, K. et al., Evaluating the Effort of Composing Design Models: A Controlled Experiment, In: MODELS'12, pages 676-691, 2012.

[7] Farias, K., Garcia, A., Whittle, J., Assessing the Impact of Aspects on Model Composition Effort, In: 9th AOSD, pages 73-84, Rennes and Saint-Malo, France, 2010.

[8] Lange, C. Assessing and Improving the Quality of Modeling A Series of Empirical Studies about the UML, PhD Thesis, Technische Universiteit Eindhoven, Eindhoven, 2007.

[9] Wust, J. The Software Design Metrics Tool for the UML, http://www.sdmetrics.com, 2012.

[10] Boehm, B., et al., Characteristics of Software Quality, vol. 1 of TRW Series of Software Technology, North-Holland Publishing Company, Amsterdam, 1978.

[11] Mccall, J., Richards, P., Walters, G. Factors in Software Quality, vol. 1-3 of AD/A-049-015/055, Springfield, 1977.

[12] Marín, B., et al., A Quality Model for Conceptual Models of MDD Environments, Advances in Soft. Engineering, 2010.

[13] Lindland, O.; Sindre, G.; Sølvberg, A. Understanding Quality in Conceptual Modeling, IEEE Software, 11(2): 42-49, March 1994.

[14] Farias, K., Garcia, A., Lucena, C. Effects of Stability on Model Composition Effort: an Exploratory Study, Journal on Software and Systems Modeling, pages 1-22, January 2013.